AD-A090 179 COLORADO STATE UNIV FORT COLLINS DEPT OF MATHEMATICS F/G 20/4
VECTORIZATION OF THE BEAM-WARMING SCHEME ON THE CRAY-1 COMPUTER--ETC(U)
AUG 80 J W THOMAS F49620-79-C-0121
UNCLASSIFIED AFOSR-TR-80-0863 NL

END
DATE FILMED
11-80
DTIC

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-80-0863	2. GOVT ACCESSION NO.: AD-A090179
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): VECTORIZATION OF THE BEAM-WARMING SCHEME ON THE CRAY-1 COMPUTER	5. TYPE OF REPORT & PERIOD COVERED: Final report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): J. W. Thomas	8. CONTRACT OR GRANT NUMBER(s): F49620-79-C-0121
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Colorado State University, Department of Mathematics, Fort Collins, CO 80523	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12. REPORT DATE: August 1980 13. NUMBER OF PAGES: Eight
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A three- dimensional version of the Beam-Warming scheme for solving the Navier-Stokes equations was implemented on the Cray-1 computer. The scheme is implicit and second-order accurate. The code is totally vectorized, allows for complicated geometries and includes a thin layer turbulence model.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AFOSR-TR- 80-0863

VECTORIZATION OF THE BEAM-WARMING SCHEME ON THE CRAY-1 COMPUTER

Final F49620-79-C-0121

1. Introduction

In recent years substantial progress has been made in the area of numerical simulation of fluid flows. However, one major limitation has been the size and speed of the available computers. Now the new generations of computers, including the ILLIAC IV, the Cray-1, and the Star-100, use vector architectures and thus offer new possibilities in the area of fluid flow simulation.

The purpose of this project was to implement the Beam-Warming scheme for solving the Navier-Stokes equations [1] on the Cray-1. More precisely, the plan was to combine the best parts of the codes written by Steger and Pulliam [2] for the NASA-Ames CDC 7600 and by Lomax and Pulliam [3] for the NASA-Ames ILLIAC IV, so as to take maximum advantage of the Cray-1. Our task was complicated by the fact that the major loops in the two codes were in opposite orders. However, we were able to capture all of the important vector operations of the ILLIAC code and to a large extent accomplish our goal.

2. Numerical Scheme

The procedure coded is an implicit finite-difference method for simulating unsteady, three-dimensional flow of a compressible, viscous fluid in arbitrary geometry. The scheme assumes that the geometry of the physical problem given in x-y-z space has been mapped onto a rectangle in the ξ-η-ζ space and that the metrics ξ_x, ξ_y, ξ_z, η_x, η_y, η_z, ζ_x, ζ_y and ζ_z and the Jacobian J of the mapping are provided. (For work on the mapping see [4] or [5].) Also, since we are interested in high Reynolds numbers flows, we use a thin layer approximation near

the rigid boundaries. (For a discussion of the turbulence model used, see [6].)

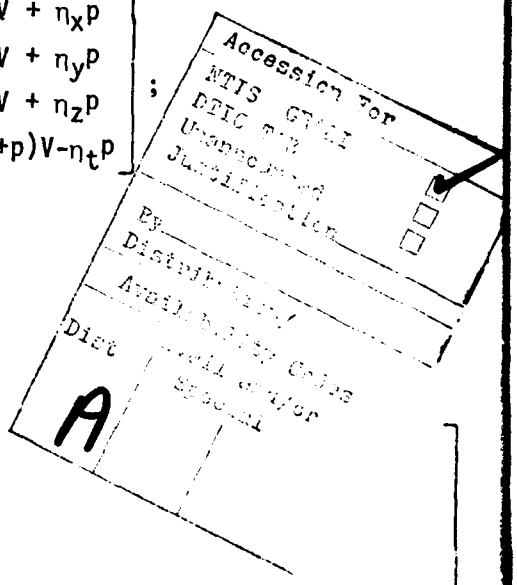
Incorporating the mapping and thin layer assumptions into the Euler equations, we are left with the following system of equations to solve

$$(1)\quad \frac{\partial \underline{q}}{\partial \tau} + \frac{\partial}{\partial \xi}(\hat{\underline{E}} - \hat{\underline{E}}_\infty) + \frac{\partial}{\partial \eta}(\hat{\underline{F}} - \hat{\underline{F}}_\infty) + \frac{\partial}{\partial \zeta}(\hat{\underline{G}} - \hat{\underline{G}}_\infty) - \frac{1}{Re}\frac{\partial \hat{\underline{S}}}{\partial \eta}$$

where

$$\underline{q} = J^{-1}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ e\end{bmatrix};\quad \hat{\underline{E}} = \begin{bmatrix}\rho U\\ \rho uU + \xi_x p\\ \rho vU + \xi_y p\\ \rho wU + \xi_z p\\ (e+p)U - \xi_t p\end{bmatrix};\quad \hat{\underline{F}} = \begin{bmatrix}\rho V\\ \rho uV + \eta_x p\\ \rho vV + \eta_y p\\ \rho wV + \eta_z p\\ (e+p)V - \eta_t p\end{bmatrix};$$

$$\underline{G} = \begin{bmatrix}\rho W\\ \rho uW + \zeta_x p\\ \rho vW + \zeta_y p\\ \rho wW + \zeta_z p\\ (e+p)W - \zeta_t p\end{bmatrix};$$



$$\underline{S} = \begin{bmatrix}0\\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)u_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_x\\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)v_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_y\\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)w_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_z\\ (\eta_x^2+\eta_y^2+\eta_z^2)[.5\mu(u^2+v^2+w^2)_\eta + \kappa Pr^{-1}(\gamma-1)^{-1}(a^2)_\eta] + (\frac{\mu}{3})(\eta_x u+\eta_y v+\eta_z w)(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\end{bmatrix};$$

$$U = \xi_t + \xi_x u + \xi_y v + \xi_z w$$

$$V = \eta_t + \eta_x u + \eta_y v + \eta_z w \quad ;$$

$$W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w$$

ρ, u, v, w and e are the density, velocity components and internal energy, respectively; ξ_x, ξ_y, ξ_z, ξ_t, η_x, ..., ζ_t are the metrics associated with the mapping; μ, κ, Pr, Re are viscosity, coefficient of thermal conductivity, Prandtl number and Reynolds number, respectively; and $\hat{E}_\infty$, $\hat{F}_\infty$ and $\hat{G}_\infty$ are the

functions $\hat{E}$, $\hat{F}$ and $\hat{G}$ evaluated at free stream values of $\hat{q}$ (this will ensure that our resulting finite difference equations will maintain a free stream flow).

Equation (1) is solved by the Beam-Warming scheme [1] or more specifically the Steger-Pulliam implementation of the Beam-Warming scheme [2]. The technique uses trapezoidal time differencing, expansion of the nonlinear terms about the n^{th} time step, and approximately factoring the resulting equation. We then obtain the following finite difference equation

$$(2)\quad (I+\frac{\Delta t}{2}\delta_\xi A^n-\varepsilon_I J^{-1}\nabla_\xi\Delta_\xi J)(I+\frac{\Delta t}{2}\delta_\eta B^n-\varepsilon_I\nabla_\eta\Delta_\eta J-\frac{\Delta t}{2Re}M^n)(I+\frac{\Delta t}{2}\delta_\zeta C^n-\varepsilon_I\nabla_\zeta\Delta_\zeta J)\Delta\underline{q}$$
$$=\Delta t(\delta_\xi\underline{\hat{E}}^n+\delta_\eta\underline{\hat{F}}^n+\delta_\zeta\underline{\hat{G}}^n-\frac{1}{Re}\delta_\eta\underline{\hat{S}}^n)-\varepsilon_E J^{-1}[(\nabla_\xi\Delta_\xi)^2+(\nabla_\eta\Delta_\eta)^2+(\nabla_\zeta\Delta_\zeta)^2]J\underline{q}^n$$

where Δt is the time increment; δ_ξ, δ_η, δ_ζ are central differences with respect to the appropriate space variable; ε_I and ε_E are amounts of dissipation added (second order and fourth order, respectively); all functions with a superscript m denote that function evaluated at time $m\Delta t$; $\Delta\underline{q} = \underline{q}^{n+1} - \underline{q}^n$; and $A = \frac{\partial\hat{E}}{\partial\underline{q}}$, $B = \frac{\partial\hat{F}}{\partial\underline{q}}$, $C = \frac{\partial\hat{G}}{\partial\underline{q}}$ and $M = \frac{\partial\hat{S}}{\partial\underline{q}}$. For an excellent paper describing the above scheme see [2].

The scheme coded is to solve equation (2). This process consists of the following four steps: (1) the explicit computation of the right-hand side; (2)-(4) the solution of three block tridiagonal systems of equations.

3. Cray-1 Code

The 7600 code to solve equation (2) is relatively straightforward (or at least as straightforward as such a large code can be). The right-hand side is evaluated and then the three block tridiagonals are

solved. An important fact is that the equation to be solved is indexed in the direction associated with that particular factor. For example, when solving the equation

$$(3) \qquad (I + \frac{\Delta t}{2}\delta_\eta B^n - \varepsilon_I \nabla_\eta \Delta_\eta J - \frac{\Delta t}{2Re} M^n)\underline{u} = \text{Right-hand side},$$

it is logical and saves storage space to fix the J and L indices (indices associated with the ξ and ζ directions, respectively) and solve the resulting system of equations indexed by K (index in the η direction). This allows the 7600 code to solve a 30-row (because of size limitations 30 is the largest number of mesh points used in any direction) block (5 x 5 blocks) tridiagonal.

Unfortunately, although the above procedure is probably vectorizable, it is nowhere near optimum for the Cray-1. The largest resulting vector would be 30 where multiples of 64 are the optimum vector sizes on the Cray-1.

Because of the vector capabilities of the ILLIAC, the ILLIAC code is quite different from the 7600 code. The ILLIAC code is written in CFD, [7], which is similar to FORTRAN so parts were able to be used almost exactly as in the ILLIAC code. However, due to peculiarities of the ILLIAC, much of the code is ILLIAC dependent (disc manipulations, manipulation of scalars, scalar arithmetic, etc.). Hence, we developed the Cray code by using the serial operation flow of the 7600 while retaining all of the vector portions of the ILLIAC code that are performed often. For another Cray-1 implementation of the ILLIAC code and the subsequent comparison, see [8].

Much of the logic in the ILLIAC code is due to the data structure and manipulation. One problem is that the ILLIAC has only 130K words

of memory so to be able to work a very large problem it is necessary to use the disc extensively. Also, two characteristics of the ILLIAC are that it will only handle vectors of length 64 or less (and it is inefficient to use less) and it is very difficult (nearly impossible) to transfer any information down the vector. Because of these limitations the data structure used in the ILLIAC code is to partition the grid into 8 x 8 x 8 blocks. A row of these blocks in a given direction is then called a pencil in that direction. Because of the space limitation of the memory, the scheme is to bring a pencil at a time into core. The solution scheme involves first sweeping through each of the x-pencils and with the data in the machine where the η-ζ directions are the components of the vector (an ξ-pencil will contain JMAX 8 x 8 η-ζ planes), calculate the necessary ξ differences in the right-hand side of equation (2). The next two steps involve doing the same thing with the η and ζ pencils. These steps are necessary with the ILLIAC since the machine will not allow arithmetic along a vector. This implementation is convenient since it makes the right-hand side calculations vector operations. The next three steps involve sweeping through the pencils in each of the directions again, at each step solving the appropriate block tridiagonal matrix. For example, equation (3) would be solved while sweeping the η-pencils. Since the left-hand side of equation (3) involves only η differences, the vector contains an 8 x 8 piece of a ξ-ζ plane. This makes solving equation (3) a perfect vector operation. We might add that for each of these sweeps the data must be in the machine in a different order (to make the operations vector and to make the matrices block tridiagonals). Hence, some of the sweeps also include the appropriate transposes to get the data in the right places. We should also add that by doing the right-hand side calculation and solving the appropriate block

tridiagonal matrix during the same sweep for the ξ and η directions, it is necessary to sweep the data four times rather than the six indicated above.

One convenient aspect about converting the ILLIAC code to the Cray was that the above described data structure is also best to use on the Cray. Initially it might seem that since the Cray vector operations have no length restrictions (except that the speed is optimum when the vector length is a multiple of 64), the best approach is to use pencils that include the entire grid or at least bigger than 8 x 8 pieces. This might be possible for problems if a disk version of the Cray code was desired. However, the above scheme for the full grid would require at least 40 x (grid size) + 50 x (largest plane size) words of storage so only relatively small problems would fit on the Cray. (If the full block tridiagonal matrix is formed, the 32 would have to be increased to 90.)

Our Cray code is written so that it is contained entirely in core. In general, the block tridiagonal solver will require 3 x (vector length) x 25 x (maximum pencil length - 2). Our block tridiagonal solver generates the matrix during the forward sweep so if 8 x 8 pencils are used, storage due to the matrix solver is minimal (64 x 25 x (pencil length +2)).

Thus we have used the same data structure for our Cray code as was used in the ILLIAC code. The storage requirements for our code are approximately 15 x (grid size) + 64 x 43 x (maximum pencil length) + 64 x 90 so at least moderately large problems can be worked using the code.

Hence, the code proceeds to solver equation (2) by sweeping the data to form the right-hand side and again sweep the data to solve the matrix equations, by the same four sweeps used in the ILLIAC code.

4. Summary

Unfortunately, at the time that it was necessary to write this report, we did not have any comparisons to present. We are presently trying to ready the code to reproduce a run common to the 7600 and ILLIAC codes. This is not the trivial matter that it should be since the run used a 30 x 30 x 21 point mesh and the Cray code was prepared to accept only multiples of 8. After the code checks out, we will proceed to run some comparisons. We will also compare running the code with our block tridiagonal solver and using a canned block diagonal solver (though the latter will require storage of the full block tridiagonal matrix).

REFERENCES

[1] Beam, Richard M. and R. F. Warming, An implicit factored scheme for the compressible Navier-Stokes equations, AIAA 3rd Comp. Fluid Dynamics Conf., Albuquerque, NM, June, 1977.

[2] Pulliam, Thomas H. and Joseph L. Steger, On implicit finite-difference simulations of three-dimensional flow, AIAA 16th Aerospace Science Meeting, Huntsville, Alabama, 1978.

[3] Pulliam, T. H. and H. Lomax, Simulation of three-dimensional compressible viscous flow on the ILLIAC IV computer, presented at the 17th Aerospace Sciences Meeting, New Orleans, LA, Jan. 1979.

[4] Thompson, Joe F., Frank C. Thames, and C. Wayne Mastin, Automatic numerical generation of a body-fitted curvilinear coordinate system for a field containing any number of arbitrary two-dimensional bodies, J. Comp. Physics, 15 (1974), 299-319.

[5] Eisenman, Peter R., A coordinate system for a viscous transonic cascade theory, J. of Comp. Physics 24 (1978), 307-338.

[6] Baldwin, B. S. and H. Lomax, Thin layer approximation and algebraic model for separated turbulent flows, presented at the AIAA 16th Aerospace Sciences Meeting, Huntsville, AL, Jan., 1978.

[7] CFD, A FORTRAN-Based Language for ILLIAC IV, Computational Fluid Dynamics Branch, NASA-Ames Research Center, Moffett Field, Calif., 1974.

[8] Buning, P. G. and J. B. Levy, Vectorization of Implicit Navier-Stokes Codes on the Cray-1 Computer, preprint.